Introduction to Scala

Practical Exercises

Collections, and the Optional[T] type

1. The code segment

   ```
   (new java.io.File(".")).listFiles
   ```

   generates a list of the contents of the current directory (or whatever directory path is supplied in the call) as instances of the type java.io.File.

   Use this as the starting point for the following questions, each of which should be answered by constructing a chain of function calls.

   a. Generate a list of the names of all the *files* in the current directory (ie. not subdirectories). Exclude hidden files (files whose names start with the '.' character).
   b. Generate two list of entries in the directory, one containing the files and the other containing the subdirectories
   c. Generate a list of Pair objects, each containing the name of a file and its size in bytes.
   d. Generate a list of the 10 smallest files in the directory, in order, together with their sizes. Do the same for the 10 largest files.
   e. Modify your answer to the above to return a Map, where the keys are the names and the values are the sizes.
   f. Generate a data structure that arranges the contents of the directory according to the first letter of their name. In other words, for the letter 'a' there should be a list of the entries whose names start with 'a', and so on…

2. Certain arithmetic operations are not defined for all possible input values. For example, divide is not defined when the divisor is 0, and square root is not defined when the input number is negative.

   Write functions for implementing versions of, for example, divide and square root, that return Option[T] values. Return Some(value) if the function is defined on the input(s), and None if not.

   Implement a further function that applies one of your functions above to each element of a Seq of inputs, generating a Seq of outputs for the valid results.